PDNO:200308559-1

METHOD AND APPARATUS FOR PERFORMING CONFLICT RESOLUTION IN DATABASE LOGGING

BY:

NITZAN PELEG
EDWARD BORTNIKOV
DROR ZERNIK

METHOD AND APPARATUS FOR PERFORMING CONFLICT RESOLUTION IN DATABASE LOGGING

BACKGROUND OF THE RELATED ART

[0001] This section is intended to introduce the reader to various aspects of art, which may be related to various aspects of the present invention that are described and/or claimed below. This discussion is believed to be helpful in providing the reader with background information to facilitate a better understanding of the various aspects of the present invention. Accordingly, it should be understood that these statements are to be read in this light, and not as admissions of prior art.

[0002] Modern computer databases may store immense amounts of data. This data is typically stored in one or more tables that comprise the database. If a database contains large amounts of data, it may take a relatively long time to perform a query to retrieve data that is of interest to a user. The time required for a database to respond to a query may have an adverse impact on the performance of the database as a whole. If the database is subject to a large number of complex queries, the response time for each query may be seriously lengthened. A query may identify a subset of elements of a table. The subset may be referred to as a "view." If a view requires information from several tables or is frequently requested by users, the view may be created as a "materialized view" to improve the performance of the database. When a view is materialized, it may actually be stored as a separate table within the database. Queries may then be run against the materialized view without incurring processing time penalties for reassembling the information contained in the materialized view each time a query that may be satisfied by the materialized view is performed.

[0003] In order to make sure that the integrity of data provided by a database is maintained, the data stored in a materialized view may need to be updated when the underlying data in the base tables that affect the materialized view is changed. When changes to underlying base tables occur, the database management system ("DBMS") may create and/or update a log showing the changes. Periodically, the DBMS may use the information contained in the log to update or refresh a materialized view.

[0004] In a complex database environment, either immediate refreshing or deferred refreshing may be employed. Immediate refreshing refers to a policy in which materialized views are updated after every change to an underlying base table. In many cases, immediate refreshing is not practical because it requires a lot of system overhead. For a deferred refresh policy, updates are collected in a log and applied periodically. Log entries may relate to either a single table entry or a range of entries. When updating a table based on the contents of the log, care must be taken to ensure that inconsistencies in the logged data do not result in the writing of erroneous data into the associated table.

[0005] Automatic range logging might introduce correctness problems to the log. There are two types of logging correctness issues that may require resolution. The first issue relates to conflicts between the range records and the second issue relates to conflicts between the single-row records and the range records. A method and apparatus that promotes effective range logging and may provide other advantages is desirable.

BRIEF DESCRIPTION OF THE DRAWINGS

[0006] Advantages of one or more disclosed embodiments may become apparent upon reading the following detailed description and upon reference to the drawings in which:

[0007] FIG. 1 is a block diagram illustrating a computer network in accordance with embodiments of the present invention;

[0008] FIG. 2 is a block diagram illustrating a materialized view underlying table update log that may be implemented in embodiments of the present invention.

[0009] FIG. 3 is a block diagram showing a system that may perform logging and conflict resolution in accordance with embodiments of the present invention;

[0010] FIG. 4 is a block diagram that illustrates conflict resolution rules in accordance with embodiments of the present invention;

[0011] FIG. 5 is a schematic diagram that illustrates a first example of duplicate elimination with ranges in accordance with embodiments of the present invention; and

[0012] FIG. 6 is a schematic diagram that illustrates a second example of duplicate elimination with ranges in accordance with embodiments of the present invention.

DETAILED DESCRIPTION

[0013] One or more specific embodiments of the present invention will be described below. In an effort to provide a concise description of these embodiments, not all features of an actual implementation are described in the specification. It should be appreciated that in the development of any such actual implementation, as in any engineering or design project, numerous implementation-specific decisions must be made to achieve the developers' specific goals, such as compliance with system-related and business-related constraints, which may vary from one implementation to another. Moreover, it should be appreciated that such a development effort might be complex and time consuming, but would nevertheless be a routine undertaking of design, fabrication, and manufacture for those of ordinary skill having the benefit of this disclosure.

[0014] Turning now to the drawings and referring initially to FIG. 1, a block diagram of a computer network architecture is illustrated and designated using a reference numeral 10. A server 20 may be connected to a plurality of client computers 22, 24 and 26. The server 20 may be connected to as many as "n" different client computers. Each client computer in the network 10 may be a functional client computer. The magnitude of "n" may be a function of the computing power or capacity of the server 20. The computing power or capacity

of the server 20 may be a function of many design factors such as the number and speed of processors and/or the size of the system memory, for example.

[0015] The server 20 may be connected via a network infrastructure 30, which may include any combination of hubs, switches, routers, and the like. While the network infrastructure 30 is illustrated as being either a local area network ("LAN"), storage area network ("SAN") a wide area network ("WAN") or a metropolitan area network ("MAN"), those skilled in the art will appreciate that the network infrastructure 30 may assume other forms or may even provide network connectivity through the Internet. As described below, the network 10 may include other servers, which may be dispersed geographically with respect to each other to support client computers in other locations.

[0016] The network infrastructure 30 may connect the server 20 to server 40, which may be representative of any other server in the network environment of server 20. The server 40 may be connected to a plurality of client computers 42, 44, and 46. As illustrated in FIG. 1, a network infrastructure 90, which may include a LAN, a WAN, a MAN or other network configuration, may be used to connect the client computers 42, 44 and 46 to the server 40. A storage device 48 such as a hard drive, storage area network ("SAN"), RAID array or the like may be attached to the server 40. The storage device 48 may be used to store a database or portion of a database for use by other network resources. Portions or partitions of a single database may be stored on various different storage devices within the network 10.

[0017] The server 40 may be adapted to create log files for updating materialized views that may be stored on the storage device 48. For example, the server 40 may be adapted to identify Insert/Update or Delete operations made to base tables that affect the materialized view and create a log entry with a timestamp indicating when the operation to the base table occurred.

[0018] The server 40 may additionally be connected to server 50, which may be connected to client computers 52 and 54. A network infrastructure 80, which may include a LAN, a WAN, a MAN or other network configuration, which may be used to connect the client computers 52, 54 to the server 50. The number of client computers connected to the servers 40 and 50 may depend on the capacity of the servers 40 and 50 to process information. A storage device 56 such as a hard drive, storage area network ("SAN"), RAID array or the like may be attached to the server 50. The storage device 56 may be used to store a database or portion of a database for use by other network resources.

[0019] The server 50 may be adapted to create log files for updating materialized views that may be stored on the storage device 56. For example, the server 50 may be adapted to identify Insert/Update or Delete operations made to base tables that affect the materialized view and create a log entry with a timestamp indicating when the operation to the base table occurred.

[0020] The server 50 may additionally be connected to the Internet 60, which may be connected to a server 70. The server 70 may be connected to a plurality of client computers 72, 74 and 76. The server 70 may be connected to as

many client computers as its computing capacity may allow. A storage device 78 such as a hard drive, storage area network ("SAN"), RAID array or the like may be attached to the server 40. The storage device 78 may be used to store a database 80 or portion of a database for use by other network resources. The database 80 may comprise a materialized view 82 (shown in dashed lines). Those of ordinary skill in the art will appreciate that other storage devices in the network 10 may store databases, which may include materialized views.

[0021] The server 70 may be adapted to create log files for updating materialized views that may be stored on the storage device 78 such as the materialized view 82. For example, the server 70 may be adapted to identify Insert/Update or Delete operations made to base tables that affect the materialized view and create a log entry with a timestamp indicating when the operation to the base table occurred.

[0022] Those of ordinary skill in the art will appreciate that the servers 20, 40, 50, and 70 may not be centrally located. Accordingly, the storage devices 48, 56 and 78 may also be at different locations. A network architecture, such as the network architecture 10, may typically result in a wide geographic distribution of computing and database resources.

[0023] The use of databases in a networked computing environment is an important tool in a modern business environment. A database may be described as a collection of related records or tuples of information or data. A relational database is a popular type of database. In a relational database, a structured set of tables or

relations is defined. The tables may be populated with rows and columns of data. The entire collection of tables makes up a relational database.

[0024] A database may be accessed through an application program, which may be referred to as a database management system or "DBMS." The DBMS typically performs database management functions. The DBMS may additionally allow users to add new data to the database or access data that is already stored in the database. An access to the database is typically referred to as a "query." A query may be performed across an entire relational database and may request data from one or more tables within the database. The organization of the data requested by a query may be called a "view." Views may not exist independently within the database, but may only exist as the output from a query.

[0025] In a networked computing environment, the information stored in a database may not all be in a centralized location. Portions of data in a single relational database may be stored on different servers on different network segments, or even in different cities or countries. To make processing the information faster, a relational database may be partitioned among a number of servers to allow parallel processing of queries. The use of materialized views may also make the processing of queries more efficient.

[0026] FIG. 2 is a block diagram illustrating a materialized view underlying table update log that may be implemented in embodiments of the present invention. When a materialized view is created, it may be designated to be

refreshed according to one of two incremental refresh policies. Those policies may be referred to as a deferred refresh policy and an immediate refresh policy.

[0027] Database tables that have one or more materialized views defined on them and that employ a deferred refresh policy may automatically maintain a log similar to the log shown in FIG. 2. The refresh operations, including reading and updating the refresh log, may be performed by a part of the DBMS that may be referred to as a refresh manager. In FIG.2, a partial excerpt of a refresh log is generally identified by the reference numeral 100. Because the refresh log contains information about Insert, Update and Delete operations, the refresh log may be referred to as an IUD log. The information shown in the log excerpt 100 is an example of the information that may be included in such a log. Those of ordinary skill in the art will appreciate that various combinations of data, including additional data or subsets of the data shown may exist in actual databases.

[0028] Each base table in a database may have its own IUD log. That log may serve all the deferred materialized views based on the base table. Accordingly, the IUD log for a particular table may be referred to as a T-log.

[0029] Each row of the IUD log 100 may constitute a separate record, which contains information about a change to the underlying table. The IUD log 100 may comprise a record column which may include an indication of the records that have been modified in a particular base table that contains data used in a particular materialized view. The record field identifies the base table row that was altered. In FIG. 2, the record column is populated with record identifiers 102a-114a, which

may be primary keys. Those of ordinary skill in the art will appreciate that the IUD log 100 may contain multiple entries to the same base table row if that row has been altered more than once. A timestamp column contains timestamps indicative of the time at which the corresponding record in the base table was inserted, updated or deleted. The timestamps are identified by the reference numerals TS 102b-TS 114b in FIG. 2. As set forth above, the timestamps may not be correctly synchronized because they may have been generated by different nodes in a networked computing environment. In such a system, various components of a base table may be distributed in partitions that are located on a number of different computing nodes, as illustrated in FIG. 1.

[0030] The IUD log 100 may include an update type column that may contain data indicative of the type of update that was performed on the base table. The update type information may be useful in determining how to update associated materialized views when a refresh operation is performed. The update type column may contain an indication that a particular entry is to be ignored, as described below. The update type column in FIG. 2 is populated with data elements identified as UT 102c-UT 114c.

[0031] The refresh log or IUD log 100 may include a column indicative of the data that was added or modified in the base table row associated with the corresponding record identifier. These data elements are identified as data 102d-data 114d in FIG. 2.

[0032] Each row or record of the IUD log 100 may also include an epoch number. In FIG. 2, the epoch numbers for the rows shown are referred to as E 102e-E 114e. The epoch number may be used to identify a group of rows or records that have been added to the IUD log 100 since a previous refresh operation was performed. A potential problem with refreshing materialized views may relate to the synchronization between the IUD log 100 and the actual performance of the refresh operation. The use of the epoch number may help to address this problem by avoiding inclusion of records corresponding to transactions that occurred outside a refresh time range or omitting records corresponding to transactions that actually occurred within a particular refresh time range.

[0033] Each base table may have a single epoch number that may be stored as part of the metadata information of the database. Metadata is data stored with the database that relates to the organizational structure and operation of the database. The epoch number may also be visible as part of the runtime information for the associated table. That information may reside in system memory during execution.

[0034] When log entries are created in the IUD log 100 (FIG. 2), the current epoch number may be read from the runtime information of the table and written to the log record. When completed, each entry in the refresh log or IUD log 100 may comprise a record that includes a record identifier (e.g Record 102a), a timestamp (e.g. TS 102b), an update type (e.g. UT 102c), a data element (Data 102d) and an epoch number (E 102e). Embodiments of the present invention may be implemented using additional items or subsets of the items listed above.

[0035] For purposes of explaining logging operations in accordance with embodiments of the present invention, consider an exact set of all the log records that are relevant to a materialized view for a given table, T. A given invocation of a refresh operation may be referred to as the table delta or T-delta. The logging to the IUD log may be done automatically as a part of the Insert/Update/Delete operations. Two types of logging may be supported, which may result in two types of records in the IUD log. The first type of record relates to IUD logging operations on single rows where each row in the base table is identified by a primary key value. A primary key is an attribute or combination of attributes that, by itself, guarantees the uniqueness of each table row. The second type of record relates to the logging of new ranges in the primary key that result from bulk inserts. A new range is a range in the primary key that was empty of data before the bulk insert happened. The new range is typically locked during the insert.

[0036] Range logging may be introduced to the system to optimize the logging time and space requirements. To reproduce the logging information from a range in the IUD log, the range and the table may be joined. This join reflects the current stage of the data in the table and, therefore, does not always reproduce the correct information. The creation and use of an IUD log to resolve conflicts is now explained with reference to FIG. 3.

[0037] FIG. 3 is a block diagram showing a system that may perform logging and conflict resolution in accordance with embodiments of the present invention. The diagram shown in FIG. 3 is generally referred to by the reference

numeral 120. A system 122 is adapted to perform logging operations and conflict resolution in accordance with embodiments of the present invention. The system 122 may operate in conjunction with a database program and may comprise a portion of such a program.

[0038] In the normal course of operation, a base or underlying table 128 may be updated when users perform IUD operations, as illustrated in FIG. 3. A materialized view 130 is based, at least in part, on the underlying table 128. Accordingly, changes to the underlying table 128 may have an impact on the materialized view 130. A logging mechanism 124 receives IUD information as the underlying table 128 is changed. The logging mechanism 124 employs the IUD information to create an IUD update log 100, as described in FIG. 2. A refresh manager 126 may be employed to periodically refresh the materialized view 130 based on the changes to the underlying table 128.

[0039] The refresh manager may obtain information from the update log 100, as shown in FIG. 3. Those of ordinary skill in the art will appreciate that the refresh manager 126 may also receive information from the underlying table itself. For example, information from the underlying table 128 may be required by the refresh manager 126 in the case of an entry corresponding to a range in the update log 100. A range entry would not typically include the actual data stored to the underlying table 128, so that data would need to be obtained from the underlying table 128 itself. When performing refresh operations on the materialized view 130, the refresh manager 126 may resolve conflicts introduced by range logging, as fully explained below.

[0040] Two separate types of logging correctness issues may require resolution. The first issue relates to conflicts between the range records. Ranges may overlap totally or partially. This conflict may happen, for example, if a user performs a bulk insert that is range-logged, then deletes a part of inserted data and performs a new bulk insert (that is also range-logged) in an overlapping area (in place of deleted data). Resolving conflicts between ranges is vital for correctness, to avoid multiple contribution to materialized views. This is an interval intersection analysis problem.

[0041] The second issue relates to conflicts between the single-row records and the range records (also called cross-type duplicates). This conflict can happen when a part of data that is inserted in bulk (and range-logged) is updated before the next invocation of refresh. Single-row records store copies of affected rows, whereas range records keep pointers into the table. Therefore, if a single-row record is logically "covered" by a range, it should not be applied, because the correct "version" of the data will be reproduced by a join between the range and the table. A double application (of the single-row record and the range) would result in duplicate contribution to materialized zed views. Therefore, the result will be incorrect.

[0042] Two algorithms are described below. The first relates to ranges conflict analysis. The second relates to range operation and single operation conflict analysis. Every time a deferred materialized view is refreshed, a new epoch may be set for all its underlying base tables. As set forth above, the epoch

value is an attribute of the base table (T.CURRENT_EPOCH). At the beginning of a refresh operation, each base table is locked, the epoch (T.CURRENT_EPOCH) is set and then the lock is released. The lock during the epoch setting guarantees that values from the same transaction will belong to the same epoch. The use of epochs may be implemented in multiple ways. For example, the epoch number may be the timestamp taken during the table lock period in the beginning of the refresh or it may be any other ever-increasing number. All the underlying base table log records that have appeared between two consecutive invocations of refresh of materialized views on that base table have the same epoch.

[0043] Every deferred materialized view maintains a vector of current epochs in the metadata area, one value per each base table. For a materialized view ("MV") that is defined on a base table T, the value MV.EPOCH [T] stands for the first epoch that was not applied to MV in T-log (i.e., the next time this materialized view will be refreshed, the T-delta computation for it should involve only those log records in T-log that have MV.EPOCH[T] <= Log_Record.EPOCH < T.CURRENT_EPOCH).

[0044] The range conflict analysis algorithm may rewrite the ranges so that they contain no overlapping ranges, but with range semantics preserved. In order to achieve this, ranges can be deleted, split, or resized.

[0045] A first axiom relating to conflict resolution may be stated as follows: if two ranges overlap, the conflict resolution is always in the favor of the younger range (i.e., the one with a greater timestamp). In other words, the range

that has been logged earlier must be changed (deleted, split, or resized) to resolve the overlap, whereas the range which has been logged later must remain intact.

[0046] The proof of the first axiom is easiest for two ranges that belong to different epochs. The range with the greater epoch value can be observed alone by some materialized view, but not vice versa. Hence, the range from the smaller epoch is at a disadvantage. The same principle also holds for two overlapping ranges that belong to the same epoch. Confilct resolution rules in accordance with embodiments of the present invention are explained below with reference to FIG. 4.

[0047] FIG. 4 is a block diagram that illustrates conflict resolution rules in accordance with embodiments of the present invention. The diagram is generally referred to by the reference numeral 200. The first example shown in FIG. 4 relates to the case when one range is contained entirely within another range. This situation is depicted in the top left column, where R_2 is entirely covered by R_1. The resulting conflict adjusted range log is depicted in the right column of FIG 3 for the case where the timestamp of R_1 is greater than R_2 (first row of the right hand column) and the case where the timestamp for R_1 is less than the timestamp for R_2 (second row of the right hand column). When the timestamp of R_1 is greater than the timestamp for R_2, the resulting range log (R_1') comprises the original range R_1. When the timestamp of R_1 is less than the timestamp for R_2, the resulting range log comprises R_1', R_2' and R_1''.

[0048] The lower row of the left hand column of FIG. 4 illustrates a case in which two ranges partially overlap. The two lower rows of the right hand column show the resulting range log components R_1' and R_2', respectively, for cases in which the timestamp of R_1 is greater than the timestamp of R_2 (third row of right hand column) and in which the timestamp of R_1 is less than the timestamp for R_2.

[0049] Following the range analysis, a range can be split into a set of subsets or fragments. Fragments produced by the range analysis can be open-ended (on bottom, top, or both). Additionally, the set can also be empty if the particular range is fully dominated by ranges with more recent timestamps.

[0050] Next, the issue of range operation and single operation conflict analysis will be discussed. The logging conflict that is relevant to this discussion is the conflict between the single-row records and the range records (also called cross-type duplicates). This conflict can happen, for example, when a part of data that is inserted in a bulk operation (and range-logged) is updated before the next invocation of the refresh operation.

[0051] Consider the case of a single log record S (that represents an Insert/Update/Delete operation) that has a primary key value in the same range as the primary key values of a log range record R. A primary key is an attribute or combination of attributes that, by itself, guarantees the uniqueness of each table row. With respect to the following discussion, a beginning range is denoted as BR and an ending range is denoted as ER. Any single row operation record in the log

is denoted as S. Any range operation record in the log is denoted as R. A primary key value of any operation is denoted as PKey.

[0052] If a single record S (that represents Insert/Update/Delete operation) has: BR.PKey <= S.PKey <= ER.Pkey, we say that record S is covered by the range. If a range covers a single-row record S, and was inserted before the operation described by S happened, it is said to screen S. A comparison between the boundaries of the range with the primary key value of the record tells whether the range covers the record. An additional comparison between the timestamps of the range and the single record tells whether the range screens the record.

[0053] Every materialized view that observes both the range and the screened record S must not apply S. This is because S did not contribute to the materialized view before the range, therefore its contribution is already counted by the join between the range and the base table. For example, if the range is followed by a deletion (and no more log record that relates to the same primary key), then the join will return a "hole" in the place of the primary key value. If, however, the range is followed by an insert operation (which means that the range did not include primary key described by the insert operation), the join will return the up-to-date data from the table. The issue of duplicate elimination with ranges is illustrated below with reference to FIG. 5.

[0054] FIG. 5 is a schematic diagram that illustrates a first example of duplicate elimination with ranges in accordance with embodiments of the present invention. The diagram is generally referred to by the reference numeral 300.

[0055] Every materialized view that has observed the range but not the screened record should take the record into account and apply it to the underlying table. Suppose that a first materialized view (MV_1) observes the log from epoch 9 (E_9) whereas a second materialized view (MV_2) observes the log from epoch 11 (E_{11}). Then, MV_1 must apply only the range but not the records screened by it, but MV_2 must apply both D_1 and I_2. A second example of duplicate elimination with ranges is discussed below with respect to FIG. 6.

[0056] FIG. 6 is a schematic diagram that illustrates a second example of duplicate elimination with ranges in accordance with embodiments of the present invention. The diagram is generally referred to by the reference numeral 400. In the example shown in FIG. 6, suppose that MV_1 observes the log from epoch 7 (E_7), MV_2 observes the log from epoch 10 (E_{10}), and third materialized view (MV_3) observes the log from epoch 11 (E_{11}). Then MV_1 must apply I_1, D_2 and the range, MV_2 must apply the range only, and MV_3 must apply D_5 and I_6.

[0057] These examples demonstrate that different materialized views apply the single-row log records selectively. Therefore, the log records that are irrelevant for some materialized views cannot be just deleted from the log, because the other materialized views might need them. Therefore, a solution is to mark the range-covered records in the way that allows selective treatment by each refresh operation. The design and correctness proof of ignore marks appears below.

[0058] A duplicate single-row record can be either irrelevant for every materialized view on the table, or for a part of the materialized views on it. The following axioms two and three discriminate between the two cases.

[0059] Axiom 2: if the latest screening range belongs to the same epoch as the single-row record, the record is irrelevant for any materialized view. In support of axiom 2, it is submitted that every materialized view observes the log on epoch boundary, hence no materialized view can observe the record separately from the range. The resolution is that the record can be deleted from the IUD log.

[0060] Axiom 3: if the single-row record belongs to epoch E, and the latest screening range belongs to epoch $E' < E$, then the record is irrelevant for each materialized view that fulfils MV.EPOCH[T] $\le E'$ and not irrelevant for each materialized view that fulfils MV.EPOCH[T] $> E'$. In support of axiom 3, it is submitted that the first type of materialized view observes the range, whereas the second type does not. The resolution is that the record will be marked using the IGNORE column: IGNORE $\leftarrow E'$. Later on, the refresh for a specific materialized view will filter the single-row records for which IGNORE >= MV.EPOCH[T] out from the log. Intuitively, the higher the IGNORE mark is, the less materialized views can apply the record. The default for the IGNORE column is 0, which means that a new (and therefore unmarked) log record is always relevant.

[0061] Referring again to the example set forth in FIG. 6, the records D_3 and I_4 must be deleted, whereas D_5 and I_6 must be marked: D_5.IGNORE = I_6.

IGNORE = E_{10}. Recalling that MV_2.EPOCH[T] = E_{10} and MV_3.EPOCH[T] = E_{11}, we get that MV_3 will apply D_5 and I_6 whereas MV_2 will not.

[0062] An ignore indication, which may be contained in the update type field of an IUD log, may receive a new value only if it already does not have a higher value. Because a record that is not deleted during the resolution can be only screened by ranges in smaller epochs, its IGNORE value is set only once. This is because some invocation of this algorithm that starts from a smaller epoch cannot discover a range that can give it a higher mark.

[0063] The following discussion relates to implementation of the IUD-log primary key. Assume EPOCH represents the epoch for single-row log records and (-epoch) represent range log records. Assume T_PKey1 through T_PKeyn represent the primary key columns (begin range for range record) of the original table. TIMESTAMP represents the logging operation timestamp. Let T_PKey denote the primary key columns of the base table. The primary key of the IUD-log may consist of the following columns, in the following order: EPOCH, T_PKey, T1MESTAMP. Notice that for range records, the EPOCH ← (-epoch) and therefore ranges and single-rows that are stored in the log can be scanned separately.

[0064] With respect to the implementation of a range conflict analysis algorithm, range conflict analysis may be performed in a single scan on the relevant ranges in the log. The ranges are read in the beginning range value order. For scan of ranges (without actual sorting of the log by (T_PKey, TIMESTAMP)),

we exploit the fact that inside each single epoch, the IUD-log records an ordered by (T_PKey, TIMESTAMP). Hence, sorting can be avoided and the required order can be achieved by activating merge-union on the ordered data in the different epochs.

[0065] The conflict analysis is typically done in the memory and the range is considered active and is typically kept in the memory only as long as its end range value is smaller or equal to the last begin range value read from the log. When a range becomes not active the result ranges are flashed to the log.

[0066] With respect to the implementation of a range operation and single operation conflict analysis, the cross-type resolution statements are performed for each range during range conflict analysis before the flash of the result to the log. The duplicate resolution rules imply two invariants for the resolution of cross-type duplicates of a range R. Every single-row record S within the boundaries of the range in the same epoch that fulfils S.TIMESTAMP > R.TIMESTAMP must be deleted (this is an efficient way to perform a single range delete per range). Every single-row record S within the range's boundaries in the greater epoch that fulfils S. IGNORE < R.EPOCH (i.e., is not screened by a later range already) must receive S.IGNORE ← R.EPOCH (this is an efficient single range update per range per epoch).

[0067] Thus, embodiments of the present invention facilitate the use of automatic range logging to optimize the logging time and space requirements. The range logging concept is expanded to include key sequenced tables as well as entry

sequenced tables. In key sequenced tables, inserts are not necessarily append operations, and may interleave with existing table rows. In range logging operations that are done only in append mode, the correctness issues are simpler. In append mode, conflicts between ranges are not possible and single operations may appear only after the insert range operations. This invention resolves correctness problems that are introduced when automatic range logging is allowed for the general case and not only in append mode.

[0068] While the invention may be susceptible to various modifications and alternative forms, specific embodiments have been shown by way of example in the drawings and will be described in detail herein. However, it should be understood that the invention is not intended to be limited to the particular forms disclosed. Rather, the invention is to cover all modifications, equivalents and alternatives falling within the spirit and scope of the invention as defined by the following appended claims.